Tedapat 4 konsep Object Oriented Programmig yaitu: polymorphism, inheritance, abstraction, encapsulation.

***Polymorphism*** diterapkan dalam bentuk ada banyak method yang bisa namanya sama bahkan constructor bisa memiliki signature (input dan tipe input yang berbeda2).

***Inheritance*** dengan kata kunci extends, dan biasanya dibuat dulu common class lalu diextend, ini untuk reusability.

***Abstraction***, ini untuk menyembunyikan kerumitan2 yang ada dan ada juga memang kelas abstract di oo

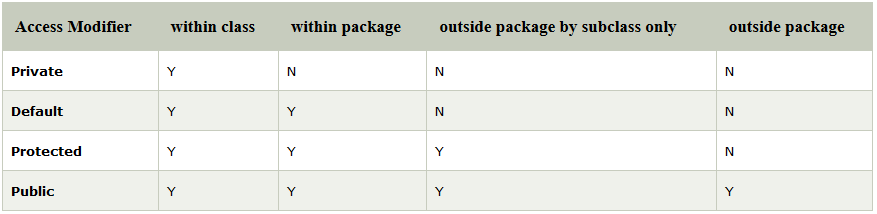
***Encapsulation*** ini juga melindungi teraksesnya properties2 yang tidak seharusnya

There are two types of modifiers in Java: **access modifiers** and **non-access modifiers**.

There are four types of Java access modifiers:

1. **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
2. **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
3. **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
4. **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

There are many non-access modifiers, such as static, abstract, synchronized, native, volatile, transient, etc. Here, we are going to learn the access modifiers only.



***Interface*** adalah class yang tidak memiliki tubuh pada method-methodnya. Method interface tersebut harus diimplementasikan dalam kelas turunannya tidak boleh tidak. Di dalam interface, deklarasi variable memiliki atribut final sehingga bersifat absolut. Keyword final inilah yang menjadi keunikan sendiri bagi interface bahwa ouput dari bagian interface berupa final yang tidak diganti pada saat implementasi kecuali di override.

Keuntungan membuat interface sendiri adalah menutupi kekurangan pada java yang hanya memperbolehkan satu kelas saja yang berhak mendapatkan warisan kelas induk (extends). Sehingga satu kelas hanya dapat menggunakan satu kelas induk, sebaliknya pada interface dapat di implementasi lebih dari satu. Ciri-ciri interface adalah interface tidak dapat di instansiasi, tidak terdapat konstruktor dan semua method interface adalah abstrak.

**String[] args.**

Tugasnya untuk menyimpan nilai argumen yang diberikan dari CMD atau terminal, sehingga kita bisa mengolahnya dalam program.

Kita akan banyak membutuhkan String[] args apabila ingin membuat program berbasis teks yang memiliki argumen.

***public*** means that the method is visible and can be called from other objects of other types. Other alternatives are private, protected, package and package-private

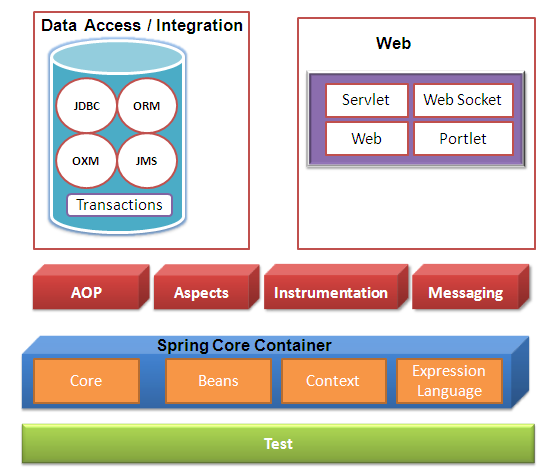
***Static*** static means that the method is associated with the class, not a specific instance (object) of that class. This means that you can call a static method without creating an object of the class.

***Void*** means that the method has no return value. If the method returned an int you would write int instead of void.

ada topik kita refactoring atau coding standar...

bagaimana sebaiknya penamaan method dilakukan ? gunakan kata kerja misalnya doCalculation, populateList, sortArray, dlsb dan gunakan camel case untuk method, dan untuk object juga. Sementara untuk class gunakan First Letter Capital

untuk boolean type gunakan kata seperti is, can. contoh isNumber, isFemale, canAbort, canIncrease, dlsb.



**Spring Core**

* It deals with Spring **IOC (Inversion of control)** or **Dependency Injection** concept.
* Spring core package is **org.springframework.core**, which provides classes and interfaces for our spring core framework development.

**Beans:**

* It deals with the **Bean factory** concept which is implemented based on **factory design pattern**.
* Bean factory package is **org.springframework.beans.factory**, which has all the classes and interfaces for implementing the inversion of control (IOC).

**Context:**

* It is mainly used to support our application objects to work with the **messaging source resources.**
* Context has **org.springframework.context** package, which has many classes and interfaces for developing the context in spring framework.

**Expression Language (EL):**

* It is an extension of Expression language in JSP technology
* EL is used for manipulating the java objects in a way of getting /setting the property values, Method invocations, Accessing the context of arrays, collections and indexes, logical and arithmetic operators, named variables, and retrieval of objects by name from spring’s IOC container.

**Aspect Oriented Layer**

This spring framework Layer contains following modules

**AOP:** It deals with Aspect Oriented Programming that allows us to use Advices, Point cuts, Method Interceptors to decouple the code.

**Aspects:**

* It provides integration with AspectJ

**Instrumentation**

* It provides support for class instrumentation and class loader implementations to be used in certain application servers

**Web Layer**

It supports us in developing the web application and it uses following components

**Servlet:**

* it is used to implement web application using the Spring MVC ( Model , View , ControllerSpring MVC separates our development code as a Model for the model objects, Web forms in the form of a View   and servlet as a controller.

**Web socket**

* Web socket communication is achieved using this module.

**Portlet:**

* It allows us to use spring MVC in the portlet environment and mimics the web-servlet module in the portlet environment.

**Web :**

* It provides us basic web – oriented applications integration features.

**Data Access and Integration Layer**

This spring framework Layer contains following modules

**JDBC:** it provides JDBC abstraction layer that helps us not to write the JDBC code.

**ORM:** it helps us in integrating all these ORM technologies that spring supports such as JPA, JDO, and Hibernate.

**OXM:** this supports Object/XML mapping implementations for JAXB, Castor, XMLBeans, JiBX and XStream.

**JMS:** this module is used for Java Messaging services inside spring framework, hence we no need to write JMS code separately

**Transaction:** it provides all the transaction API’s for Transaction management inside spring framework.

**Test Layer**

* It helps us to **test the spring components** with JUnit or TestNG tools.

Hibernate is Object Relational Mapping (ORM), to map object in java application to table database.

**AuthenticationServer** that returns a JWT token that we can use for further requests against our secured services. Thanks to **Spring Security,** we have secured our MicroServices so that the rest of the API can only be called with a valid JWT token.  
  
JSON Web Token (**JWT**)

The **JWT token** is like a key of a house. If the key is lost or stolen, you have to have a new key made or change the lock. This also applies to our JWT Token. If the token should fall into the wrong hands, it is imperative that the secret **(application.yml)** is exchanged at both microservices.

Another possibility is to keep the **ExpireDate** of the JWT token as low as possible. This would cause the stolen JWT token to become invalid in a very short time.

kunci di git hanya hal2 berikut ini :

1. ada yang menggunakan user interface gui

2. ada yang menggunakan line command

masing2 sebenarnya sama saja

ada

1. git pull ini untuk ambil dari repository lalu tarik ke lokal kita

2. setelah lakukan perubahan lalu kita mau push, lakukan git add

3. git add akan meminta list perubahan apa saja sebagai catatan, meski sebenarnya git secara sistem sudah mentrack apa saja perubahan di repo yang kita pull tapi akan lebih mudah untuk mentrack jika kita beri catatan perubahan

misal contoh catatan perubahan yang bisa kita buat :

a. Ecko merubah kelas controller Mahasiswa, terjadi penambahan karena tipe constructor dari kelas Model yang dibuat Andi sudah berbeda

b. dst....

4. git push nah ini baru lah di push code kita ke repo utama

4 ini siklus sederhana git

ada juga perintah berikutnya:

5. git diff ini mau cek apa saja perubahan atau perbedaan versi kode kita, dan ini lbih mudah digunakan pada git versi GUI ...

6. .gitignore ... ini adalah file... ketika kita git push atau git pull, daftar file di gitignore ini akan di-ignore. fungsinya apa ?

.gitignore ini menyimpan daftar list file2 yang sering kita gunakan untuk testing, atau file tipe besar agar tidak didownload ulang dan membuat repo jadi besar, misalnya file gambar jika kita bekerja pada spring web mvc, misalnya juga file library, agar didownload di sisi kita sendiri....

dalam git ada yg disebut branch dsb ... nah git branch ya seperti cabang...

anggap kode utama kita adalah main. lalu ketika kita mau lakukan pull, sebaiknya dari main kita branch dulu agar kode utama tidak terganggu.... dan programmer lain juga bisa membranch dari main ... sehingga banyak branch yang bisa ditarik keluar dan dikerjakan bersama2

biasanya, lead programmer akan mengecek smua branch yang di push oleh member programmernya